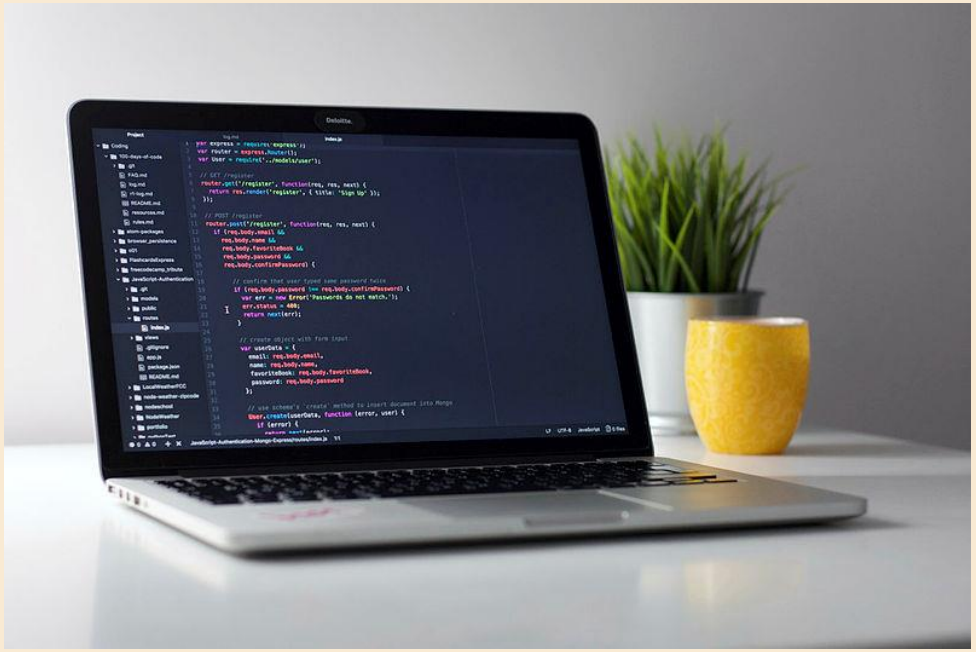


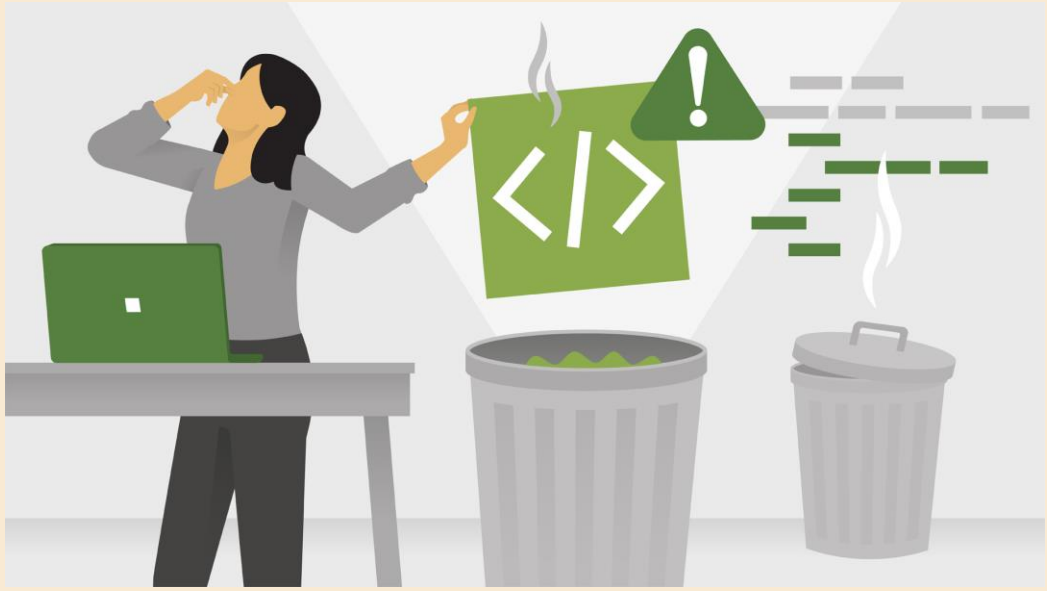
CODE SMELLS

Testdriven Development - Running

Rudi Stene

bouvet

Code smells

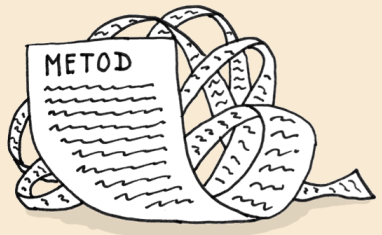


https://cdn.lynda.com/course/2812496/2812496-1568408298845-16x9.jpg

- Maximize Cohesion
  - Cohesion is a metric telling how strongly related and coherent are the responsibilities within the classes of an application
- Minimize Coupling
  - Coupling is a metric for measuring the degree of interdependence between the classes of an application

bouvet

Bloaters



https://sourcemaking.com/images/refactoring-illustrations/2x/long-method-1.png?id=d94fbb3889bb52815774

- **Long method**

  More than ten lines should make you start asking questions. Method should only do one thing and do it well.

- **Long class**

  A class contains many fields/methods/lines of code. No more than 50 lines.
  Class should have only one responsibility.

- **Long parameter list**

  More than three or four parameters for a method. Can indicate coupling violation.

Bloaters

- ## Data clumps

  Different parts of the code contain identical groups of variables. clumps should be turned into their own classes.

  Example: Position class instead of repeating usage of x and y parameters

- ## Primitive obsession

  Use of primitives instead of small objects for simple tasks.

  Example: EmailAddress, PhoneNumber

https://jfiaffe.files.wordpress.com/2015/06/the-old-vs-the-new.jpg

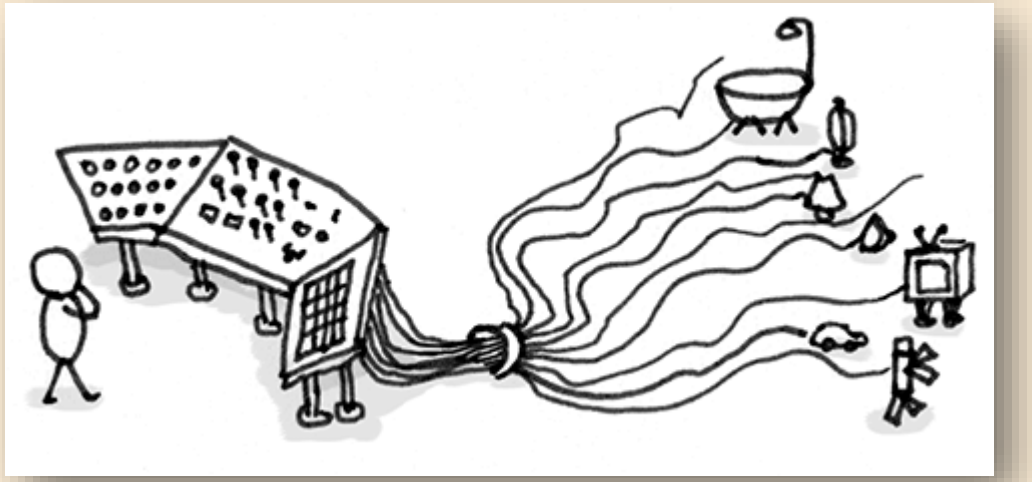
Object-orientation abusers

- **Switch statements**

  You have a complex `switch` operator or sequence of `if` statements.

- **Temporary fields**

  Class contains an instance variable set only in certain circumstances.



https://sourcemaking.com/images/refactoring-illustrations/switch-statements-1.png?id=7bed925679187c2802ed

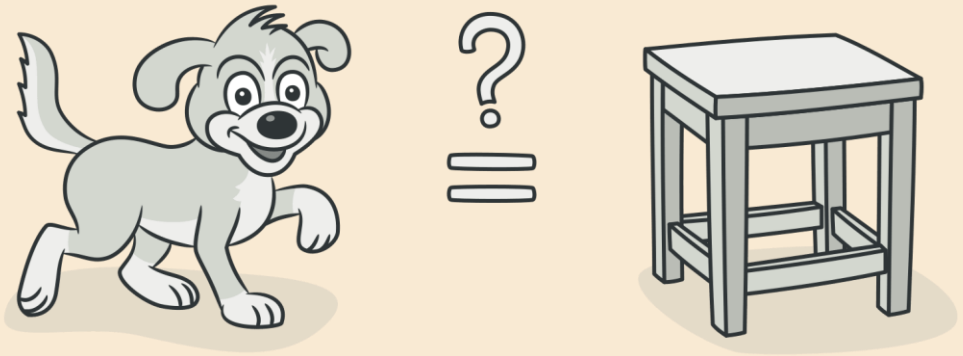
Object-orientation abusers

- **Refused bequest**

  When a subclass uses only some of the methods and properties inherited from its parents.

- **Alternative Classes with Different Interfaces**

  If two classes are similar on the inside, but different on the outside, perhaps they can be modified to share a common interface..



https://refactoring.guru/images/refactoring/content/smells/refused-bequest-01-3x.png

Change preventers

- **Divergent change**

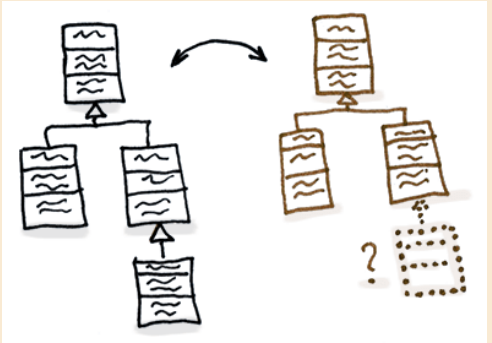
  When one class is commonly changed in different ways for different reasons.

- **Shotgun surgery**

  One change, forces lots of little changes in different classes.

- **Parallel Inheritance Hierarchies**

  Special case of shotgun surgery. Creating a subclass of one class, forces subclass of another.



https://sourcemaking.com/images/refactoring-illustrations/parallel-inheritance-hierarchies-1.png?id=6db459493612332d77db

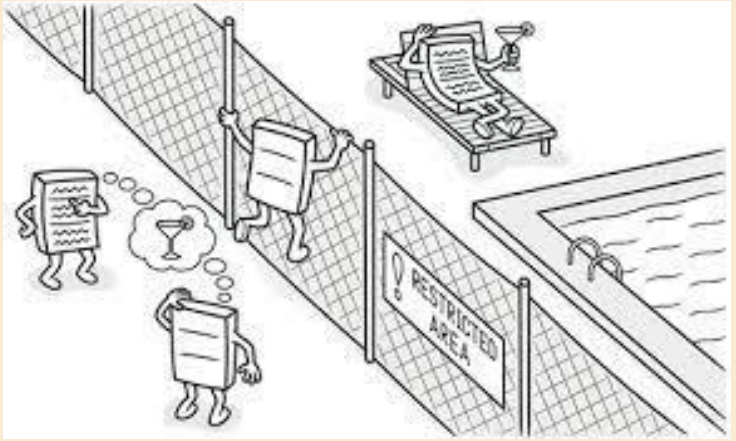
Couplers

- **Feature envy**

  A class that uses public methods or fields of another class excessively.

- **Inappropriate intimacy**

  A class that uses protected or internal methods or fields of another class excessively. Special case of feature envy especially with inheritance.



https://refactoring.guru/images/refactoring/content/smells/feature-envy-01-3x.png

Couplers

- ## Message chains

  Too many dots per line.

  `Dog.Body.Tail.Wag()` **instead of** `Dog.ExpressHappiness()`

- ## Middle man

  If a class is delegating all its work, cut out the middleman.

Dispensables

- **Comments**

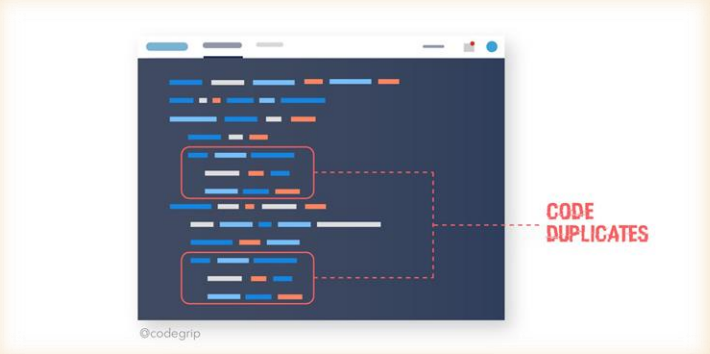
  Make effort to create code that expresses intent instead of comments.

- **Duplicate code**

  Two code fragments look almost identical.

- **Lazy class**

  A class that does too little. May be acting only as middle man or a data class or can be caused by speculative generality



https://assets.codegrip.tech/wp-content/uploads/2019/10/03143434/image1.jpg

Dispensables

- **Data class**

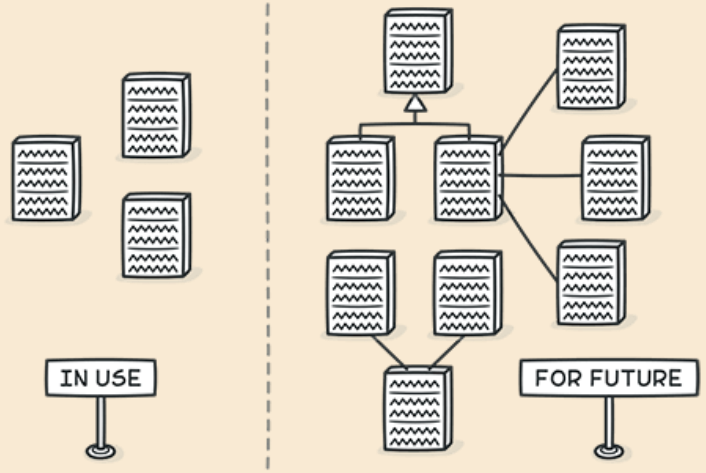
  Class holding only fields, properties, but no logic.

- **Dead code**

  A variable, parameter, field, method or class is no longer used

- **Speculative generality**

  There is an unused class, method, field or parameter.



https://refactoring.guru/images/refactoring/content/smells/speculative-generality-01.png?id=c804fce5c6c5c34b4d93

Any questions?

Thanks for your attention

Contact information:

rudi.stene@bouvet.no

rudistene